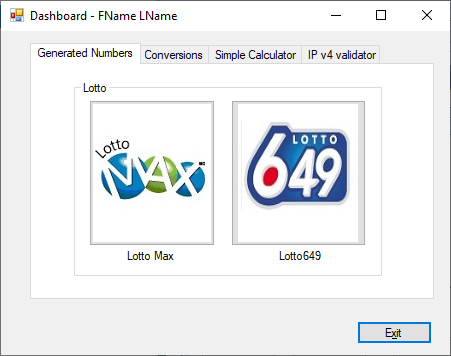
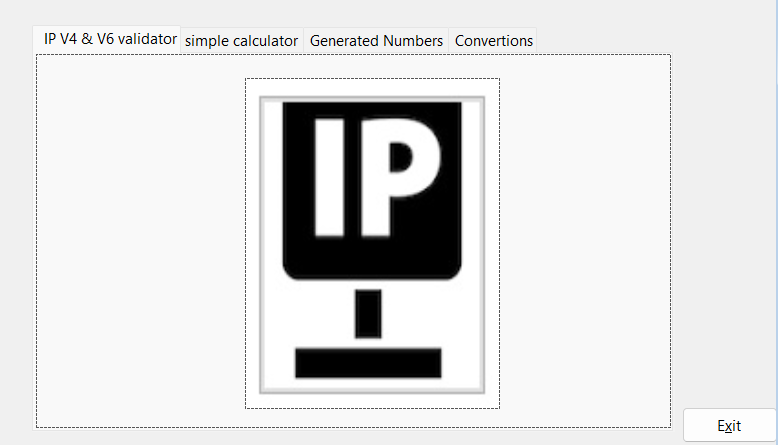
|  |
| --- |
| College LaSalle |
| Project - Oriented Object Programming User and Technical Manual |
|  |
| Presented to: Mihai Maftei. |

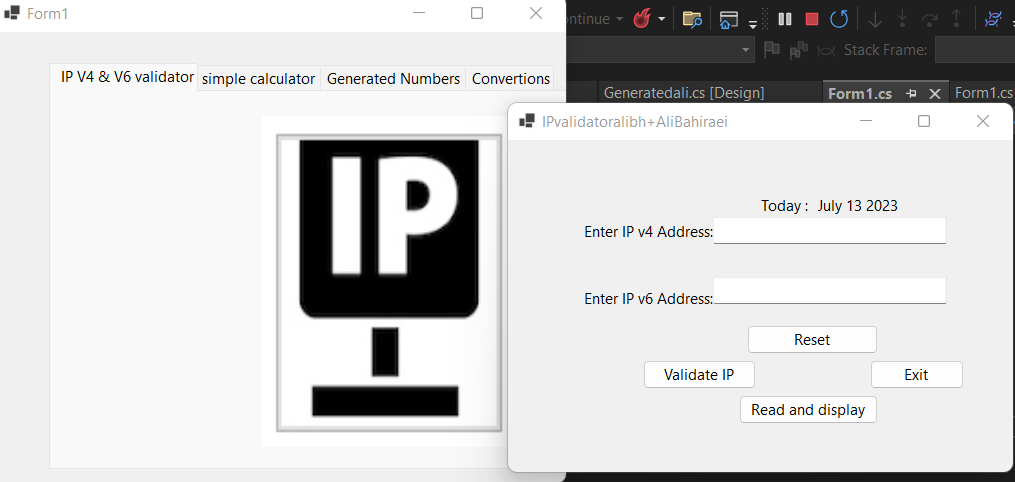
|  |
| --- |
| Your name: Ali Bahiraei  11/7/2023  Version 1.0 |

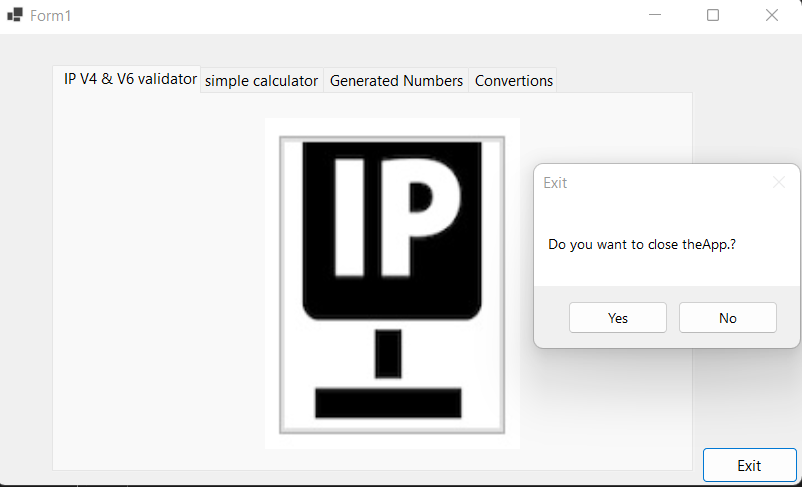
1. **Start by adding a short description of your project, and the languages (technologies) used:**
2. Language C#
3. I used visual Studio version 2022
4. **Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).**



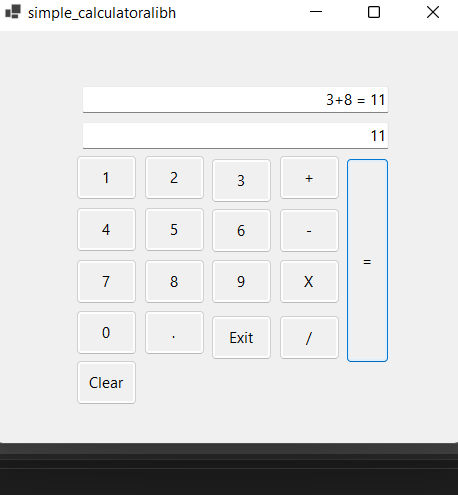
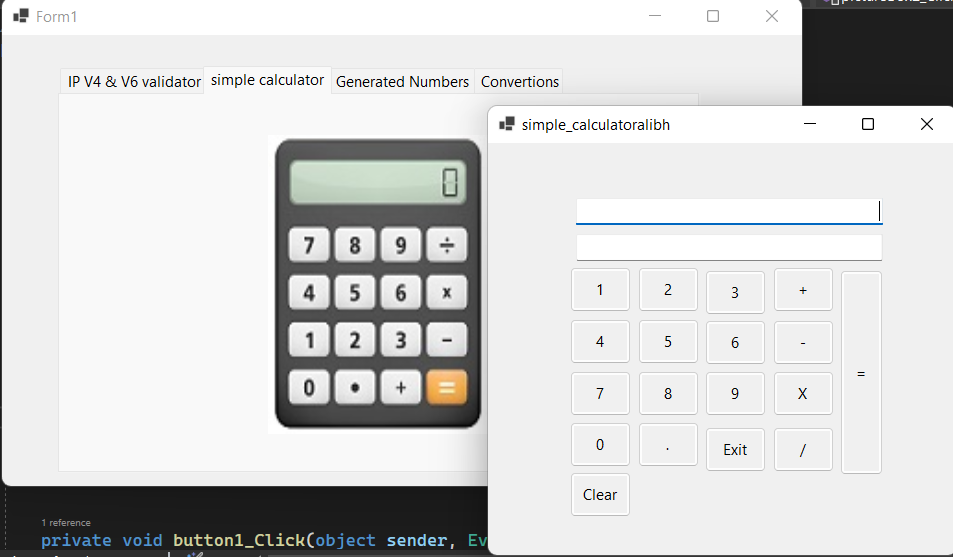
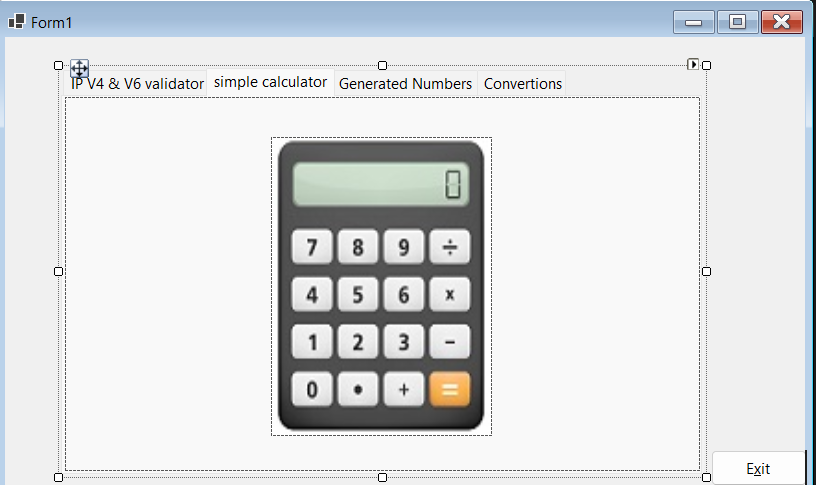
1. If you click on tab we can see some project that we can use it



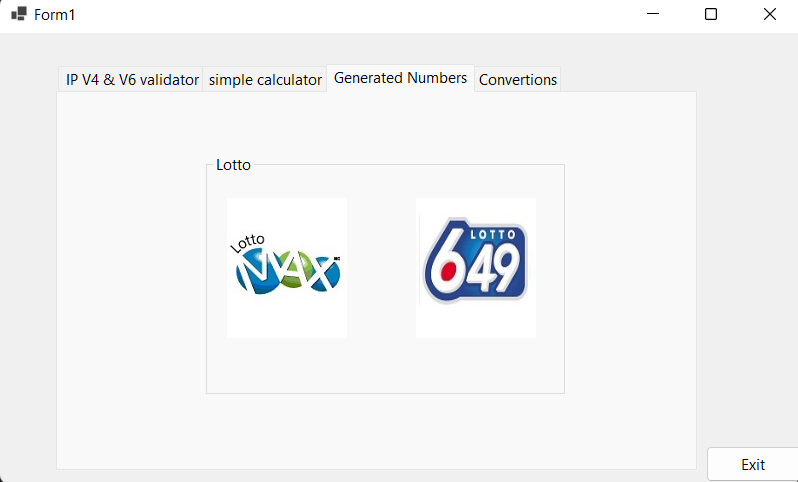
1. If you click on tab it means we can open separate project and use it
2. 
3. …
4. If you click on the Exit button it shows one message that we sure leave application if we sure press yes or no continue

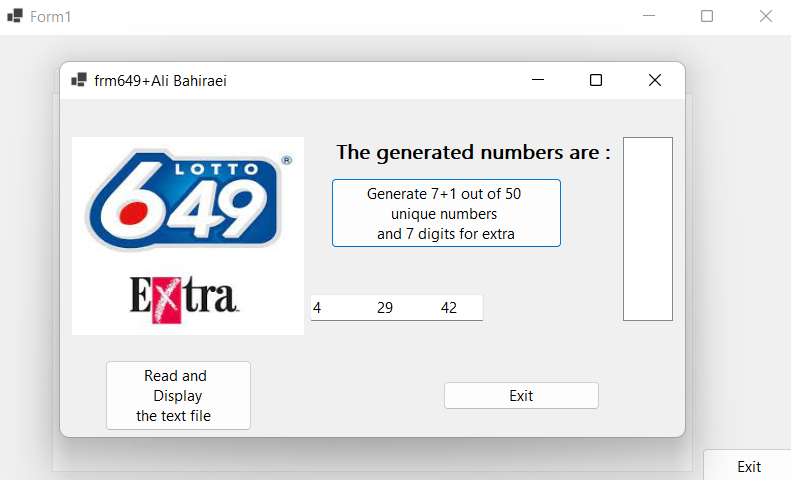


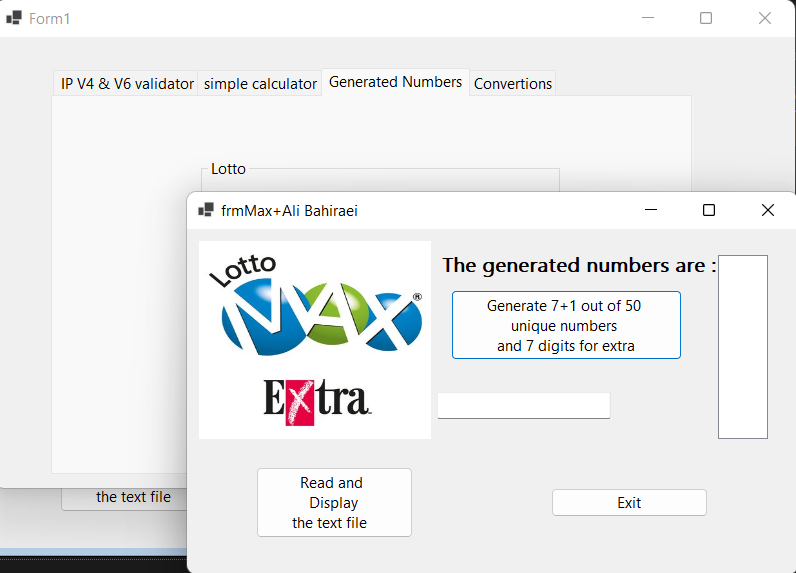
Calculator:



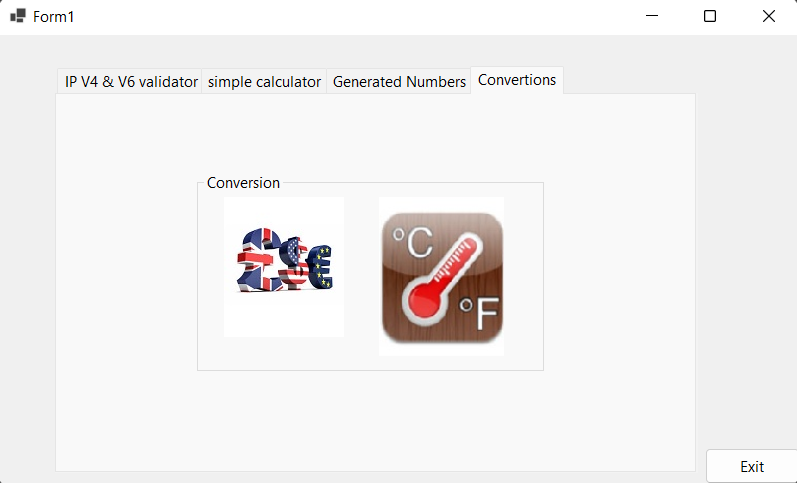
Lotto:(frm649)(frmMax)

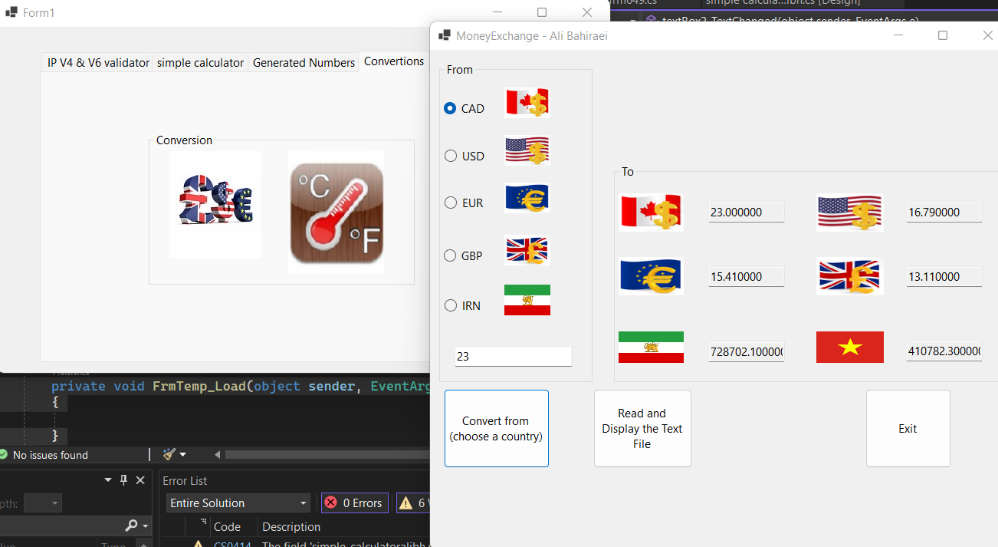




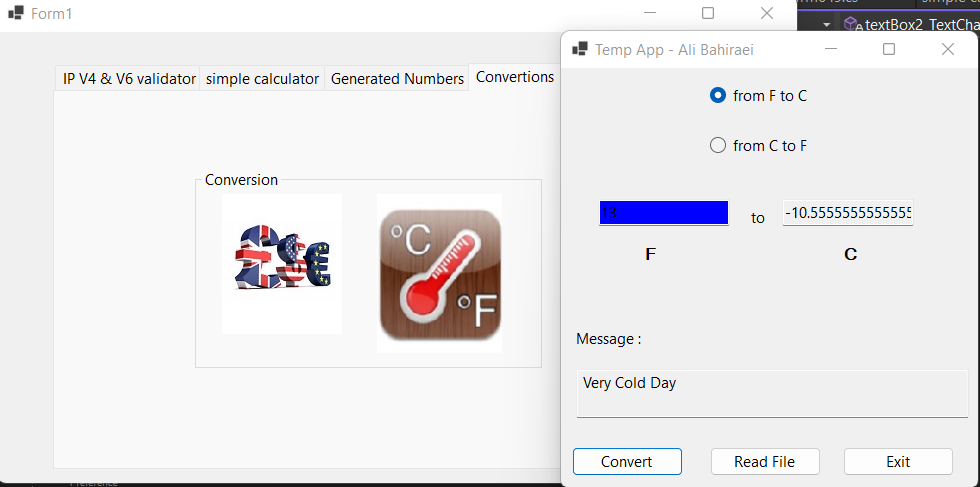


***FrmMoney+Ali Bahiraei:***

******

******

***frmTemp+Ali Bahiraei:***

******

1. **Present the code of your application (forms).**

LotoMAX

The form contains three buttons:

"Generate Numbers" (button1): When this button is clicked, the application generates a list of numbers from 1 to 50 and shuffles them randomly. It then selects the first 8 numbers from the shuffled list and stores them in a string. The current date and time are recorded along with the lottery name ("Max"), and this information is combined into a line to be saved in a text file called "LottoNbrsMax.txt". The generated numbers are displayed in a textbox for the user to see.

"View Numbers" (button2): Clicking this button reads the contents of the "LottoNbrsMax.txt" file and displays them in a message box. The message box shows the previously saved lottery numbers, along with the date and time when the numbers were generated.

"Exit" (button3): When this button is clicked, the application asks the user if they want to close the app. If the user confirms, the form is closed.

StreamWriter:

The StreamWriter class is used to write text data to a file.

In the code, it is used to save the generated lottery numbers and other relevant information to a text file.

The StreamWriter is initialized with the path of the file where the data will be saved (filePath).

After generating the lottery numbers and creating the line to be saved, the StreamWriter is used to write the line to the file using the WriteLine() method.

Finally, the StreamWriter is closed to release the file resources.

… //frmMax

namespace project\_section1

{

public partial class frmMax : Form

{

public frmMax()

{

InitializeComponent();

}

private void lottoali2\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

List<int> numbersList = Enumerable.Range(1, 50).ToList(); // Creates a list of numbers 1 to 50

Random random = new Random();

string numbers = "";

// Shuffle my numbers

for (int i = numbersList.Count - 1; i > 0; i--)

{

int j = random.Next(i + 1);

int temp = numbersList[i];

numbersList[i] = numbersList[j];

numbersList[j] = temp;

}

// Determine the number of random numbers to generate

int frmMax = 8;

int numNumbers = frmMax; // same thing for 649

// Add the first 8 to the shuffled numbers to the string

for (int i = 0; i < 8; i++)

{

numbers += numbersList[i].ToString() + "\t";

}

// Remove the trailing comma

numbers = numbers.TrimEnd(',');

// Create the line to be saved in the text file

string lotteryName = "Max"; // we will do the same for 649

string dateTime = DateTime.Now.ToString("yyyy/M/d h:mm:ss tt");

string line = $"{lotteryName}, {dateTime}, {numbers}";

string filePath = @".\Files\LottoNbrsMax.txt";

// Save the line to the text file

StreamWriter writer = new StreamWriter(filePath, true);

writer.WriteLine(line);

writer.Close();

textBox1.Text = numbers;

}

private void button2\_Click(object sender, EventArgs e)

{

// Read the contents of the text file

string fileContent;

string filePath = @".\Files\LottoNbrsMax.txt";

StreamReader reader = new StreamReader(filePath);

fileContent = reader.ReadToEnd();

reader.Close();

// Display the contents in a message box

string message = $"Lottery Numbers:\n\n{fileContent}";

string title = "Lotto Numbers - Ali Bahiraei";

MessageBox.Show(message, title);

}

private void button3\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to close theApp.? ", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

}

}

Loto649

*The form contains three buttons, just like in the previous application:*

*"Generate Numbers" (button1): When this button is clicked, the application generates a list of numbers from 1 to 49 and shuffles them randomly. It then selects the first 7 numbers from the shuffled list and stores them in a string. The current date and time are recorded along with the lottery name ("649"), and this information is combined into a line to be saved in a text file called "LottoNbrs649.txt". The generated numbers are displayed in a textbox for the user to see.*

*"View Numbers" (button2): Clicking this button reads the contents of the "LottoNbrs649.txt" file and displays them in a message box. The message box shows the previously saved lottery numbers for the "649" game, along with the date and time when the numbers were generated.*

*"Exit" (button3): When this button is clicked, the application asks the user if they want to close the app. If the user confirms, the form is closed.*

*Similar to the "Max" lottery application, this "649" lottery application demonstrates the basics of creating a simple Windows Forms utility for generating and recording random lottery numbers. It allows users to quickly get their lucky numbers for the "649" game and review past lottery entries. However, like the previous application, it lacks additional features such as saving numbers for different dates, supporting multiple lottery games, or incorporating advanced functionalities like checking for winning numbers.*

… //frm649

namespace project\_section1

{

public partial class frm649 : Form

{

public frm649()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

List<int> numbersList = Enumerable.Range(1, 49).ToList();

Random random = new Random();

string numbers = "";

for (int i = numbersList.Count - 1; i > 0; i--)

{

int j = random.Next(i + 1);

int temp = numbersList[i];

numbersList[i] = numbersList[j];

numbersList[j] = temp;

}

int frm649 = 8;

int numNumbers = frm649;

for (int i = 0; i < 7; i++)

{

numbers += numbersList[i].ToString() + "\t";

}

numbers = numbers.TrimEnd(',');

string lotteryName = "649";

string dateTime = DateTime.Now.ToString("yyyy/M/d h:mm:ss tt");

string line = $"{lotteryName}, {dateTime}, {numbers}";

string filePath = @".\Files\LottoNbrs649.txt";

Directory.CreateDirectory(Path.GetDirectoryName(filePath));

StreamWriter writer = new StreamWriter(filePath, true);

writer.WriteLine(line);

writer.Close();

textBox1.Text = numbers;

}

private void button2\_Click(object sender, EventArgs e)

{

string fileContent;

string filePath = @".\Files\LottoNbrs649.txt";

StreamReader reader = new StreamReader(filePath);

fileContent = reader.ReadToEnd();

reader.Close();

string message = $"Lottery Numbers:\n\n{fileContent}";

string title = "Lotto Numbers - Ali Bahiraei";

MessageBox.Show(message, title);

}

private void button3\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to close theApp.? ", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void Generatedali\_Load(object sender, EventArgs e)

{

}

}

}

IPv4

… using System.Data;

using System.Drawing;

using System.Linq;

using System.Net;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace project\_section1

{

//Name:Ali Bahiraei

//Date:11/07/2023

//description:used regex for validate ip address and date & use ismatch method it's continue for validating.

public partial class IPvalidatoralibh : Form

{

FileStream mhb = null;

public IPvalidatoralibh()

{

InitializeComponent();

}

string dir = @".\Files\";

string pathBin = @".\Files\IPv4IPv6Bin.txt";

private void label1\_Click(object sender, EventArgs e)

{

}

private void ValidateIP\_Click(object sender, EventArgs e)

{

string ipAddress = v4address.Text;

string ipAddress2 = v6address.Text;

// Regular expression pattern for IP address validation

string pattern = @"^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";

// Validate the IP address using regular expression

bool isValid = Regex.IsMatch(ipAddress, pattern);

if (isValid)

{

MessageBox.Show(ipAddress + "\nThe IP is correct.", "Valid IP", MessageBoxButtons.OK, MessageBoxIcon.Information);

try

{

FileStream fs = new FileStream(pathBin, FileMode.Append, FileAccess.Write);

BinaryWriter bw = new BinaryWriter(fs);

bw.Write(ipAddress);

bw.Write(DateTime.Now.ToString("yyyy/MM/dd H;mm;ss tt"));

bw.Close();

}

catch (Exception ex)

{

MessageBox.Show($"{ex.Message}", "Error");

}

finally { if (mhb != null) mhb.Close(); }

}

else

{

MessageBox.Show("Invalid IP address.", "Validation Result", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

bool isValid2 = Regex.IsMatch(ipAddress2, pattern);

if (isValid2)

{

MessageBox.Show("IP address is valid.", "Validation Result", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Invalid IP address.", "Validation Result", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Reset\_Click(object sender, EventArgs e)

{

v4address.Clear();

v6address.Clear();

}

private void Exit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to exit?", "Exit", MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

this.Close();

}

private void IPvalidatoralibh\_Load(object sender, EventArgs e)

{

lbldate.Text = DateTime.Now.ToString("MMMM dd yyyy");

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

using (FileStream fileStream = new FileStream(pathBin, FileMode.Open, FileAccess.Read))

{

using (BinaryReader bR = new BinaryReader(fileStream))

{

string textToPrint = "";

while (bR.PeekChar() != -1)

{

string IPContent = bR.ReadString();

textToPrint += IPContent + "\n";// IPDate + "\n";

}

MessageBox.Show(textToPrint);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while reading the file: {ex.Message}", "Error");

}

}

}

private bool IsValidData(string data)

{

string ipAddress = v4address.Text;

string ipAddress2 = v6address.Text;

// Regular expression pattern for IP address validation

string pattern = @"^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";

bool isValid = Regex.IsMatch(ipAddress, pattern);

bool isValid2 = Regex.IsMatch(ipAddress2, pattern);

return !string.IsNullOrEmpty(data);

}

}

}

***Calculator:*** ***The Calculator class is the core component of the application and handles the arithmetic calculations. It maintains three private decimal variables: currentValue, operand1, and operand2, and a string variable op to store the operator (+, -, \*, or /) used for the calculations. The CurrentValue property allows access to the current result of the calculator.***

***The class provides methods for each arithmetic operation (Add, Subtract, Multiply, Divide), which are called when the respective operator buttons are clicked in the user interface. Each method sets the first operand (operand1) to the value displayed in the calculator's textbox and updates the currentValue accordingly.***

***When the "Equals" button is clicked, the Equals method is invoked. It performs the appropriate operation based on the selected operator and the second operand (operand2). The result is then displayed in the second textbox (textBox2).***

***The user interface is designed using Windows Forms components, with buttons for each digit, operator, and "Equals," as well as a textbox to display the input and output. The code utilizes event handlers to capture button clicks and updates the textbox accordingly. The textBox1 is used for entering the arithmetic expression, and textBox2 displays the final result.***

***The application also handles edge cases such as division by zero, throwing a DivideByZeroException when necessary.***

***Overall, this simple calculator demonstrates the basics of building a graphical calculator application in C#. It provides a straightforward and user-friendly interface for performing basic arithmetic operations, making it suitable for simple calculations and quick tasks. However, it lacks more advanced features like parentheses support, memory functions, or scientific calculations, which might be desired in more comprehensive calculator applications. Nonetheless, it serves as a useful starting point for beginners learning C# and graphical application development.***

//project section 2

public class Calculator

{

private decimal currentValue;

private decimal operand1;

private decimal operand2;

private string op;

public Calculator()

{

currentValue = 0;

operand1 = 0;

operand2 = 0;

op = null;

}

public decimal CurrentValue

{

get { return currentValue; }

}

public void Add(decimal displayValue)

{

operand1 = displayValue;

currentValue = operand1;

op = "+";

}

public void Subtract(decimal displayValue)

{

operand1 = displayValue;

currentValue = operand1;

op = "-";

}

public void Multiply(decimal displayValue)

{

operand1 = displayValue;

currentValue = operand1;

op = "\*";

}

public void Divide(decimal displayValue)

{

operand1 = displayValue;

currentValue = operand1;

op = "/";

}

public void Equals()

{

switch (op)

{

case "+":

currentValue += operand2;

break;

case "-":

currentValue -= operand2;

break;

case "\*":

currentValue \*= operand2;

break;

case "/":

if (operand2 != 0)

{

currentValue /= operand2;

}

else

{

throw new DivideByZeroException("Cannot divide by zero.");

}

break;

}

}

public void Equals(decimal displayValue)

{

operand2 = displayValue;

Equals();

}

public void Clear()

{

currentValue = 0;

operand1 = 0;

operand2 = 0;

op = null;

}private void button14\_Click(object sender, EventArgs e)

{

string input = textBox1.Text;

if (input.Contains("+") || input.Contains("-") || input.Contains("X") || input.Contains("/"))

{

try

{

// Split the input into operands and operator

string[] parts = input.Split(new char[] { '+', '-', 'X', '/' }, StringSplitOptions.RemoveEmptyEntries);

if (parts.Length == 2)

{

decimal operand1 = decimal.Parse(parts[0]);

decimal operand2 = decimal.Parse(parts[1]);

decimal result = 0;

// Determine the operator and perform the corresponding operation

if (input.Contains("+"))

{

calculator.Add(operand1);

calculator.Equals(operand2);

}

else if (input.Contains("-"))

{

calculator.Subtract(operand1);

calculator.Equals(operand2);

}

else if (input.Contains("X"))

{

calculator.Multiply(operand1);

calculator.Equals(operand2);

}

else if (input.Contains("/"))

{

calculator.Divide(operand1);

calculator.Equals(operand2);

}

textBox2.Text = calculator.CurrentValue.ToString();

textBox1.Text = input + " = " + calculator.CurrentValue.ToString();

}

else

{

MessageBox.Show("Invalid input. Please enter a valid arithmetic expression.");

}

}

catch (Exception ex)

{

MessageBox.Show("Error occurred during evaluation: " + ex.Message);

}

}

else

{

MessageBox.Show("No arithmetic operation found in the input.");

}

***FrmMoney+Ali Bahiraei:***

***The provided code represents a Money Conversion Calculator application using a Windows Forms interface. The application allows users to convert monetary values between different currencies, including CAD (Canadian Dollar), USD (US Dollar), EUR (Euro), GBP (British Pound), and IRR (Iranian Rial). It also provides conversions to VND (Vietnamese Dong).***

***User Input: The application allows users to enter a numeric value representing the amount to be converted in a text box.***

*Currency Selection: Users can choose the currency they wish to convert from using radio buttons for CAD, USD, EUR, GBP, or IRR.*

*Conversion Logic: When the "Convert" button is clicked, the application checks the user input for valid numeric input using regular expressions. It then performs the currency conversion calculations based on the selected currency and displays the converted amounts in corresponding text boxes.*

*Output Display: The application shows the converted amounts in CAD, USD, EUR, GBP, IRR, and VND (Vietnamese Dong) formats.*

*Logging: Each conversion is logged with a timestamp, input value, and converted amounts, and the log is appended to a text file named "MoneyConversions.txt" located in the "./Files/" directory.*

*Reading the Log: The application allows users to read and display the conversion log from the text file by clicking the "Read Display Text File" button.*

*Application Timing: The application keeps track of the elapsed time since it was started. If the user decides to exit the application by clicking the "Quit" button, a confirmation message box displays the elapsed time, and the user can choose to proceed with quitting or continue using the application.*

//Ali Bahiraei

//cash convertor

//Date:25/07/2023

namespace Project\_Final

{

public partial class FrmMoney : Form

{

private string conversionFilePath = @".\Files\MoneyConversions.txt";

Stopwatch elapsedTime = new Stopwatch();

public FrmMoney()

{

InitializeComponent();

elapsedTime.Start();

}

private void btn\_Cnvr\_Click(object sender, EventArgs e)

{

try

{

Regex regex = new Regex(@"^\d\*\.?\d\*$");

if (!regex.IsMatch(textBox\_inputValue.Text))

{

MessageBox.Show("Input must be a number.", "Invalid input", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

double inputValue = double.Parse(textBox\_inputValue.Text);

double cadToUsd = 0.73;

double cadToEur = 0.67;

double cadToGbp = 0.57;

double cadToIrr = 31682.7;

double cadToVnd = 17860.1;

double usdToCad = 1.36;

double usdToEur = 0.91;

double usdToGbp = 0.76;

double usdToIrr = 42105.0;

double usdToVnd = 23680.3;

double eurToCad = 1.49;

double eurToUsd = 1.10;

double eurToGbp = 0.84;

double eurToIrr = 46830.7;

double eurToVnd = 26058;

double gbpToCad = 1.76;

double gbpToUsd = 1.31;

double gbpToEur = 1.19;

double gbpToIrr = 55805.7;

double gbpToVnd = 30550.1;

double irrToCad = 0.000031;

double irrToUsd = 0.000024;

double irrToEur = 0.000021;

double irrToGbp = 0.000018;

double irrToVnd = 0.55676;

double usd, cad, eur, gbp, irr, vnd;

if (radioButtonCAD.Checked)

{

cad = inputValue;

usd = cad \* cadToUsd;

eur = cad \* cadToEur;

gbp = cad \* cadToGbp;

irr = cad \* cadToIrr;

vnd = cad \* cadToVnd;

}

else if (radioButtonUSD.Checked)

{

usd = inputValue;

cad = usd \* usdToCad;

eur = usd \* usdToEur;

gbp = usd \* usdToGbp;

irr = usd \* usdToIrr;

vnd = usd \* usdToVnd;

}

else if (radioButtonEUR.Checked)

{

eur = inputValue;

cad = eur \* eurToCad;

usd = eur \* eurToUsd;

gbp = eur \* eurToGbp;

irr = eur \* eurToIrr;

vnd = eur \* eurToVnd;

}

else if (radioButtonGBP.Checked)

{

gbp = inputValue;

cad = gbp \* gbpToCad;

usd = gbp \* gbpToUsd;

eur = gbp \* gbpToEur;

irr = gbp \* gbpToIrr;

vnd = gbp \* gbpToVnd;

}

else // radioButtonIRN.Checked

{

irr = inputValue;

cad = irr \* irrToCad;

usd = irr \* irrToUsd;

eur = irr \* irrToEur;

gbp = irr \* irrToGbp;

vnd = irr \* irrToVnd;

}

textBox\_CAD.Text = cad.ToString("F6");

textBox\_USD.Text = usd.ToString("F6");

textBox\_EUR.Text = eur.ToString("F6");

textBox\_GBP.Text = gbp.ToString("F6");

textBox\_IRN.Text = irr.ToString("F6");

textBox\_VND.Text = vnd.ToString("F6");

string conversion = $"{DateTime.Now}\n{inputValue:F6} input = {cad:F6} CAD; {usd:F6} USD; {eur:F6} EUR; {gbp:F6} GBP; {irr:F6} IRR; {vnd:F6} VND;\n"; // this line is modified

StreamWriter sw = new StreamWriter(conversionFilePath, true);

sw.Write(conversion);

sw.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Conversion Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btn\_ReadDisplayTextFile\_Click(object sender, EventArgs e)

{

try

{

StreamReader sr = new StreamReader(conversionFilePath);

MessageBox.Show(sr.ReadToEnd(), "Money Conversions - Erfan", MessageBoxButtons.OK, MessageBoxIcon.Information);

sr.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "File Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void radioButtonCAD\_CheckedChanged(object sender, EventArgs e)

{

}

private void radioButtonUSD\_CheckedChanged(object sender, EventArgs e)

{

//radiobuttonUSD

}

private void radioButtonEUR\_CheckedChanged(object sender, EventArgs e)

{

//radioEUR

}

private void radioButtonGBP\_CheckedChanged(object sender, EventArgs e)

{

//radiobuttonGBP

}

private void radioButtonIRN\_CheckedChanged(object sender, EventArgs e)

{

//radioIRN

}

private void textBox\_inputValue\_TextChanged(object sender, EventArgs e)

{

//inputvalue

}

private void textBox\_CAD\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_EUR\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_IRN\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_USD\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_GBP\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_VND\_TextChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

elapsedTime.Stop();

TimeSpan ts = elapsedTime.Elapsed;

string elapsedTimeFormatted = String.Format("{0:00} minutes, {1:00} seconds",

ts.Minutes, ts.Seconds);

DialogResult dialogResult = MessageBox.Show($"Do you want to quit this app? You have been here for {elapsedTimeFormatted}.",

"Exit Application",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (dialogResult == DialogResult.Yes)

{

this.Close();

}

else

{

elapsedTime.Start();

}

}

private void FrmMoney\_Load(object sender, EventArgs e)

{

}

}

}

***Convertor+Tmp:***

***The provided code represents a Temperature Conversion Application using a Windows Forms interface. The application allows users to convert temperatures between Fahrenheit and Celsius and provides temperature descriptions based on the entered value. It also logs each conversion with timestamps and descriptions to a text file named "TempConversions.txt" located in the "./Files/" directory.***

***User Input: The application provides a text box for users to enter a numeric temperature value.***

* *Conversion Logic: When the "Convert" button is clicked, the application checks the user input for valid numeric input using regular expressions. It then performs temperature conversions between Fahrenheit and Celsius based on the selected radio button: Fahrenheit to Celsius (°F to °C) or Celsius to Fahrenheit (°C to °F).*
* *Temperature Description: The application provides temperature descriptions based on the converted value. The descriptions range from "Water boils" to "Extremely Cold Day" for both Celsius and Fahrenheit.*
* *Output Display: The converted temperature value is displayed in a text box, along with the corresponding temperature description.*
* *Color Coding: The application changes the background color of the Celsius text box based on the converted temperature value. Red indicates a temperature above 30°C, green for temperatures between 10°C and 30°C, and blue for temperatures below 10°C.*
* *Logging: Each conversion is logged with a timestamp, input value, converted value, and temperature description, and the log is appended to a text file named "TempConversions.txt."*
* *Reading the Log: The application allows users to read and display the conversion log from the text file by clicking the "Read File" button.*
* *User Interface: The user interface of the Temperature Conversion Application is simple and user-friendly. It includes input fields for entering temperature values, radio buttons for selecting the conversion type, buttons for conversion and reading the log, and text boxes to display the converted values and descriptions.*

public partial class FrmTemp : Form

{

private string conversionFilePath = @".\Files\TempConversions.txt";

public FrmTemp()

{

InitializeComponent();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void btn\_cnvr\_Click(object sender, EventArgs e)

{

try

{

string temp = textBox\_C.Text;

if (string.IsNullOrEmpty(temp)) throw new Exception("Please enter a temperature");

// REGEX

if (!Regex.IsMatch(temp, @"^-?\d\*\.?\d\*$")) throw new Exception("Please enter a valid number");

double tempValue = double.Parse(temp);

StreamWriter sw = new StreamWriter(conversionFilePath, true);

string description = ""; // initialize the variable

if (radioButton\_CF.Checked)

{

double result = (tempValue - 32) \* 5 / 9;

textBox2.Text = result.ToString();

description = GetDescription(result);

sw.WriteLine($"{temp} F = {result} C, {DateTime.Now} {description}");

textBox\_C.BackColor = GetColor(result);

}

else if (radioButton\_FC.Checked)

{

//textBox\_C.BackColor = GetColorF(tempValue);

double result = tempValue \* 9 / 5 + 32;

textBox2.Text = result.ToString();

description = GetDescriptionF(result);

sw.WriteLine($"{temp} C = {result} F, {DateTime.Now} {description}");

textBox\_C.BackColor = GetColorF(result);

}

textBox\_message.Text = description;

sw.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btn\_ReadFile\_Click(object sender, EventArgs e)

{

try

{

using (StreamReader sr = new StreamReader(conversionFilePath))

{

string content = sr.ReadToEnd();

MessageBox.Show(content, "Temperature Conversions - Erfan");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btn\_Exit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to close theApp.? ", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void radioButton\_CF\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton\_CF.Checked)

{

// Get the current positions of the labels

Point labelFPosition = Label\_F.Location;

Point labelCPosition = Label\_C.Location;

// Swap the positions of the labels

Label\_F.Location = labelCPosition;

Label\_C.Location = labelFPosition;

}

}

private void radioButton\_FC\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton\_FC.Checked)

{

// Get the current positions of the labels

Point labelFPosition = Label\_F.Location;

Point labelCPosition = Label\_C.Location;

// Swap the positions of the labels

Label\_F.Location = labelCPosition;

Label\_C.Location = labelFPosition;

}

}

private string GetDescription(double temp)

{

if (temp >= 100)

return "Water boils";

else if (temp >= 40)

return "Hot Bath";

else if (temp >= 37)

return "Body temperature";

else if (temp >= 30)

return "Beach weather";

else if (temp >= 21)

return "Room temperature";

else if (temp >= 10)

return "Cool Day";

else if (temp >= 0)

return "Freezing point of water";

else if (temp >= -18)

return "Very Cold Day";

else

return "Extremely Cold Day";

}

private Color GetColor(double temp)

{

if (temp >= 30)

return Color.Red;

else if (temp >= 10)

return Color.Green;

else

return Color.Blue;

}

private Color GetColorF(double temp)

{

if (temp >= 86)

return Color.Red;

else if (temp >= 55)

return Color.Green;

else

return Color.Blue;

}

private string GetDescriptionF(double temp)

{

if (temp >= 212)

return "Water boils";

else if (temp >= 104)

return "Hot Bath";

else if (temp >= 98.6)

return "Body temperature";

else if (temp >= 86)

return "Beach weather";

else if (temp >= 70)

return "Room temperature";

else if (temp >= 50)

return "Cool Day";

else if (temp >= 32)

return "Freezing point of water";

else if (temp >= 0)

return "Very Cold Day";

else

return "Extremely Cold Day";

}

private void FrmTemp\_Load(object sender, EventArgs e)

{

}

}

}

1. **Present the classes and/or methods that you create or you did use in the project.**

***frmMoney***

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
| 1. Class Calc   2. Class FrmTemp | Enter a short description of the class or of the method I use class in console for defining for operator like add etc and then I use clear class for clearing data I use class in the class I use switch case for operator .  2.a Windows Forms form for temperature conversion. |
| 1. public partial class simple\_calculatoralibh : Form | This is the main class that represents the Windows Forms application. It inherits from the Form class and serves as a simple calculator GUI. |
| 1. public partial class frm649 : Form | This is the main class that represents the Windows Forms application. It inherits from the Form class. |
| 1. Stopwatch() | This class is used for measuring elapsed time. An instance of this class is created as elapsedTime and used to track the time spent in the application. |
| 1. Regex() | This class is used for working with regular expressions. It is used to validate the input in the btn\_Cnvr\_Click method. |
| 1. StreamWriter() | This class is used to write text to a file. It is used in the btn\_Cnvr\_Click method to log conversion data to the file. |
| 1. StreamReader() | This class is used to read text from a file. It is used in the btn\_ReadDisplayTextFile\_Click method to read and display the conversion data from the file. |
| 1. public partial class FrmTemp : Form | This class is a Windows Forms class and inherits from the Form class. It represents the main form of the application for temperature conversion. |
| 1. FrmTemp() | class contains the logic for temperature conversion, event handlers for various controls on the form, and private helper methods (GetDescription, GetColor, GetColorF, and GetDescriptionF) used for determining temperature descriptions and colors. |
| 1. Class FileStream   , BinaryWriter,  BinaryReader  Stream writer :  Stream reader :  File stream : | Enter a short description:  -@".\Files\IPv4IPv6Bin.txt";actually I have one main class that is public and it’s name is IPvalidator , binarywriter for reading code as binary.actually is a struct but we used it for providing a date.  - To read data from binary file  -that simplifies writing text data to files, providing an easy and convenient way to manage file output operations.  - It provides an easy and efficient way to handle file input operations, allowing developers to read and process textual content from files with ease.  -for reading and writing binary data to files at a low-level. |
| 1. button2\_Click, button3\_Click, button6\_Click, button5\_Click, button4\_Click, button9\_Click, button8\_Click, button7\_Click, button15\_Click, button16\_Click, button10\_Click, button11\_Click, button12\_Click, button13\_Click, button14\_Click, button18\_Click, button17\_Click   calculator | These are event handler methods for various button clicks on the calculator form. Each method appends the corresponding character or symbol to the text box (textBox1). |
| 1. **Methods**   bool IsValidData(string data) | -I used class private like this for validating data for IPV4 & IPV6 |
| 1. PeekChar() | that allows you to preview the next character in the input stream without actually consuming it |
| Method public FrmTemp() | Constructor method for the FrmTemp class |
| 1. **FrmMoney()** | Constructor method for the FrmMoney class. It initializes the form and starts the elapsedTime stopwatch. |
| 1. **radioButtonCAD\_CheckedChanged(object sender, EventArgs e)** | Event handler for the checked changed event of the CAD radio button (radioButtonCAD). |
| 1. **radioButtonUSD\_CheckedChanged(object sender, EventArgs e)** 2. **.** 3. **..** 4. **.** 5. **.** 6. **.** | Event handler for the checked changed event of the USD radio button (radioButtonUSD) |
| **4.Method private string GetDescription(double temp)** | Returns a description based on the provided temperature value. |
| **5.** **Method private Color GetColor(double temp)** | Returns a Color value based on the provided temperature value. |
| **button1\_Click(object sender, EventArgs e)** | Event handler for the click event of button1. This method generates random lottery numbers for the 649 lottery game and writes them to a text file. It also displays the generated numbers in textBox1. |
| **button2\_Click(object sender, EventArgs e)** | Event handler for the click event of button2. This method reads the content of the text file containing lottery numbers and displays it in a message box. |
| **button3\_Click(object sender, EventArgs e)** | Event handler for the click event of button3. This method displays a message box to confirm if the user wants to close the application. If the user selects "Yes," the application is closed. |
| **6.Method private string GetDescriptionF(double temp)** | This method use for Returns a description based on the provided temperature value |
| 1. Constructor() : public Calculator()   {  currentValue = 0;  operand1 = 0;  operand2 = 0;  op = null; }  public frmMax(): | Enter a short description of the class or of the method I use one constructor for returning current value I use get method it means read a value .  Constructor method used for initializing the form. |

1. **Present the difficulties that you have, what was the hardest and the easiest part of your project.**

**Project one :IP address**

Actually for first part it means ip address my difficult part was read and display bc we need to create file to save data so we need create (dir) in whole project that we use in code and another parts like validate reset and exit were easy bc for validate just I use regex and put condition and for reset and exit just I used last code .

15/07/2023(calculator):actually my problem was how to do another operated with last result like regular calculator . because for class when I used class it was so difficult to handle another operator but I defined another variable for third operator.

18/07/2023(calculator):I managed to organized data and I didn’t have any problem.

17/07/2023(lottomax): it was easy section but actually I didn’t know why I used input text because we don’t need text box but in addition it was a little hard how to use read streamer file streamer or something like that but after I check this description I understood how to use it

25/07/2023(FrmTemp): Code Reusability:

Currently, the temperature descriptions and color-coding logic are hardcoded in multiple methods. A more modular approach could be adopted, such as storing the temperature descriptions and color-coding rules in data structures or configuration files. This would enhance code reusability and make it easier to update or expand the application with new temperature ranges and descriptions in the future.

User Interface Design:

While the current user interface is functional, it lacks some visual enhancements and interactivity. Improving the user interface design, adding labels, tooltips, and clearer instructions for users, and making it more intuitive would enhance the overall user experience.

25/07/2023(frmMoney):

File I/O and Exception Handling:

The code uses file I/O operations to read and write conversion data to a text file. While this is a straightforward approach, it lacks robust exception handling and error recovery. Incorporating more robust error handling for file operations and ensuring graceful recovery from file-related issues would improve the application's stability.

Unit Tests:

The code does not include any unit tests to validate the correctness of the currency conversion logic. Implementing unit tests to verify the accuracy of conversion calculations would help identify and address any potential bugs or errors.

StreamReader: This is a .NET class that allows you to read characters from a stream (like a file) in a specific encoding.

filePath: This is a string variable that holds the path to the file you want to read.

**StreamReader reader** = new StreamReader(filePath);: Here, a new instance of the StreamReader class is created with the filePath as the argument. This initializes the reader object to read the file specified by filePath.

**fileContent** = reader.ReadToEnd();: The ReadToEnd() method of the StreamReader class is used to read the entire content of the file and store it in the fileContent variable. It reads from the current position in the stream until the end of the file.